NomadGS

OpenEmbedded Overview

Matthew Locke

CTO

Nomad Global Solutions, Inc.

Agenda

- Open Source build systems
- What is OpenEmbedded?
- OpenEmbedded Internals
- Summary
- OpenEmbedded in Open Test Lab
- Discussion

Background

- Developers of the Open Test Lab (OTL)
  - Open Embedded experience from this project.
- Long history in embedded Linux
  - Everything from wireless access points to mobile phones to robots.
- Current focus is mainlining features needed for Linux on mobile phones and tools for integrating open source technology into product development.
  - Power Management, process definition, build systems, OTL

Open Source build systems

- Build from source
  - Buildroot
  - Embedded Gentoo
  - Linux From Scratch
  - **OpenEmbedded**
- Build from binaries
  - Makefile/RPM (Fedora)

BUILD FROM SCRATCH!!

OpenEmbedded - What is it?

- A completely self contained cross build system for embedded devices
- Uses a build from scratch methodology
- Collection of meta data that describe how to build:
  - Over a thousand packages including bootloaders, libraries, and applications
  - For ~60 target machines including the a780, N770 and x86
  - Over 40 package/machine configurations (distributions)
- Does not include source code. Automatically downloads source using metadata.
- Does provide ability to add patches to build

Open Embedded - methodology

- Two parts to Open Embedded
  - **Bitbake** - "Task executor". Cmd line tool that is the interface to the OpenEmbedded build system. Processes the metadata and executes the instructions contained in the metadata.
  - **Metadata** - the recipes that describe how to setup the build environment, build packages and create distro images.
- The build environment components are downloaded and built from source.
  - Components are toolchain, quilt, autotools and a few other utilities
  - Removes any dependency on host OS - works on any Linux distro and MacOS X.
- Downloads latest source code
  - However, provides ability to lock packages to specific version
- Metadata is broken into two high level types - configuration and package files.

Metadata structure - Configuration files

- Configuration files define how the build environment is setup, package versions, information, global inheritance, target boards, final image configuration.
- Four types of configuration files
  - **Distro** - highest level configuration which defines:
    - Toolchain and package versions
    - Package configuration - xserver can be built in several configurations. Distro defines which configuration is built.
    - Sets Distro information variables
    - High level settings such as use udev for device nodes and final image format.
  - **Machines** - sets information needed to build a kernel for specific target boards
    - Defines the kernel package that builds for the target board
    - Defines architecture name
    - Kernel cmd line settings
    - Root file system type and creation settings
    - Additional package dependencies

Metadata structure - Configuration files

- **Local** - bitbake configuration settings for your builds
  - Output image formats (jffs2, tar)
  - Location of source - local tarballs or cvs repositories
  - Override package configurations (toolchain, xserver etc)
- **Bitbake** - configuration settings for how bitbake processes metadata
  - URLs of source code
  - Cmds to download (cvs, wget)
  - Patch applications
- Include files
  - Both configuration files and bb files support include directive
  - Often used for package version defintions that are similar in metaimages

Metadata structure - Package files

- Bitbake recipe files (.bb)
  - Contain the necessary environment variables, cmds and steps need to build a package
  - Similar steps to other meta data packaging systems such as RPM, Debian.
  - Do_stage(), do_configure(), do_compile(), do_install(), etc.
- Three types .bb files
  - **Classes** - contains common steps for a class of packages.
    - For example, all kernel builds have make, make install, make modules.
  - **Packages** - inherits classes and adds or overrides package specific settings and steps.
    - For example uboot requires a header be placed on the kernel image. The board specific package file will add this additional step.
  - **Metaimage** - defines the collection of packages to be used by the distros using RECOMMENDS and DEPENDS statements. Sets image specific variables.

Open Embedded - Setup

- Getting started instructions on are http://www.openembedded.org
- Be prepared for large start up cost
- Obtain bitbake
  - Bitbake is stored in a svn repository
  - Need svn on your host
  - Bitbake is a python script so no compile necessary
- Obtain metadata - two choices
  - Metadata is stored in a monotone repository
    - Need to get jam and boost libraries to compile and setup monotone
  - Use metadata snapshot
    - May be out of date and not work for your desired configuration
- Need approximately 5GB free diskspace
- Remember everything is downloaded so that will take time

Open Embedded - Usage

- Getting started instructions on are http://www.openembedded.org
- Setup local configuration file to select your distro and machine and any other options you require.
- Bitbake <meta image name>

Open Embedded - Summary

- Very powerful metadata system
- Many, many packages already supported
- Can build anything from a complete PDA to a DVR to a wireless access point software distro
  - Maemo, Familiar, MythTV, unSlung
- Start up cost can be high
- Metadata learning curve is high
- No support for building apps outside the OE system
- Fairly large open source community using it and maintaining it
- Finding a version of metadata that "just works" can be a challenge

Open Embedded and Open Test Lab

- Chose OE for several reasons:
  - Many, many packages already supported
  - Build system ready to use
  - Large community behind it
- OTL wraps OE to connect into the web UI
- OTL software takes the user selected parameters and auto generates a configuration file
- We archive the built environment, images and source code snapshot. This was required for an efficient automated build environment.
- OTL takes the kernel and root filesystem images and installs them on the target
  - Tftp for the kernel
  - Nfs for the root filesystem.

Discussion

NomadGS

Matthew Locke

CTO

matt@nomadgs.com

Nomad Global Solutions, Inc.